SENSORED USER MANUAL
SECTION 1 - GROUP H

SENSORED

In this manual, we will be going through the steps needed to run the model. This project is intended to be done using Amazon Sagemaker and other services to see end result.

NOTE: *The services we used here are not free of charge and we would like to put beforehand that every resource you use will be billable.*

STEP 1.

In this step, we will see how to set up a notebook instance using Sagemaker.

Start by logging in to Amazon Console, opening the Sagemaker dashboard. Go to the side panel and click on the Notebook dropdown menu to create a **Notebook Instance.** Then click on **Create Notebook Instance** and follow the steps below:

1. **Notebook Instance Setting:** In this section, you may choose the notebook instance name first. Then for the **Notebook Instance Type,** even though by default **ml.t2.medium** is available, it won't let us train our model. Therefore select one that is suited for training like **ml.m5.xlarge**. **NOTE**: Training instance types are only allowed upon request via the customer service console. This will take up to 48 hours to approve.
2. **Permission and Encryption:** In this menu under **IAM role,** click on **Create a new role.** You should get a pop-up dialog box, where you have to select **Any S3 bucket** to get access to the training data we uploaded priorly. Click on **Create role** after finishing.
3. **Git repositories:** Here you will clone the *https://github.com/samuel-Kurabachew/spoiler_alert* repository to the current notebook instance.

You are done with setting up the notebook instance. It will take a while to download and process the necessary files and environment. Have a cup of coffee in the meantime :).

When the process is done, click on **Open Jupyter** and follow along with the notebook. When finished with your work, don't forget to stop the notebook instance to avoid any incurring costs. This can be achieved by clicking on the radio button before the instance name and then click on **Action** then **Stop** from the dropdown menu.

STEP 2.

In this step, we will create Lambda function.

However, before we create this Lambda function we need an **IAM Role** for the Lambda Function. This will allow our Lambda function to have the necessary permission to make a call to Sagemaker.

1. **Create an IAM Role**: Using the **AWS Console**, navigate to the **IAM** page and click on **Roles**. Then, click on **Create role**. Make sure that the **AWS service** is the type of trusted entity selected and choose **Lambda** as the service that will use this role, then click **Next: Permissions.** In the search box type sagemaker and select the check box next to the **AmazonSageMakerFullAccess** policy. Then, click on **Next: Review.** Lastly, give this role a name. Make sure you use a name that you will remember later on, for example *LambdaSpoilerAlertRole*. Then, click on **Create role**.
2. **Create Lambda Function**: Using the **AWS Console**, navigate to the **AWS Lambda** page and click on **Create a function**. When you get to the next page, make sure that **Author from scratch** is selected. Now, name your Lambda function, using a name that you will remember later on, for example, *spoiler_alert_func*. Make sure that the **Python 3.6** runtime is selected and then choose the role that you created in the previous part. Then, click on **Create Function.** On the next page, you will see some information about the Lambda function you've just created. If you scroll down you should see an editor in which you can write the code that will be executed when your Lambda function is triggered. You will find the code in GitHub repository here with file name **lambda_script.py** *https://github.com/samuel-Kurabachew/spoiler_alert*

Now we are done with creating the Lambda function, the next step is to set up the API gateway our web app will use to communicate with the model through the Lambda function.

STEP 3.

Setting up API Gateway.

1. Using **AWS Console**, navigate to **Amazon API Gateway** and then click on **Get started.**
2. On the next page, make sure that **New API** is selected and give the new API a name, for example, *spoiler_alert_api*. Then, click on **Create API**.
3. Select the **Actions** dropdown menu and click **Create Method**. A new blank method will be created, select its **dropdown menu** and select **POST**, then click on the **checkmark** beside it.
4. For the **integration point**, make sure that **Lambda Function** is selected and click on the **Use Lambda Proxy integration**.
5. Type the name of the Lambda function you created earlier into the Lambda Function text entry box and then click on **Save**. Click on **OK** in the pop-up box that then appears, giving permission to API Gateway to invoke the Lambda function you created.
6. The last step in creating the API Gateway is to select the **Action**s dropdown and click on **Deploy API**. You will need to create a new Deployment stage and name it anything you like, for example, *test*.

You have now successfully set up a public API to access the SageMaker model. From this page, a URL is found at the top of the page highlighted in blue which will be needed when using the simple web interface we created. Make sure to copy this. Next, download the **index.html** file from the git repository and open it with a text editor of your choice. Paste the URL you copied before to the line which says **REPLACE WITH YOUR PUBLIC URL**. Voila! You are done. Now you should be able to send a review through the web app and see whether it is a spoiler or not. Make sure the model is deployed and running to see the result.

After finishing your test, Please delete any Lambda functions and API endpoints created to avoid unexpected bills. It will keep adding up even though you are not using the services. The same goes for the notebook instances and deployed endpoints.